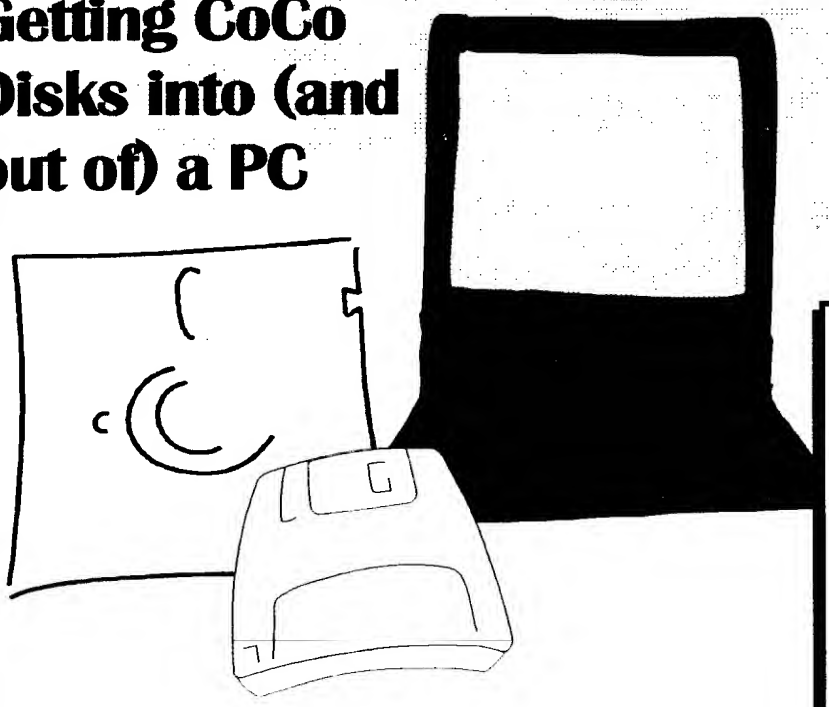
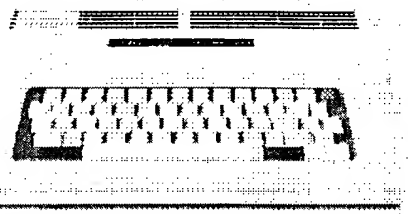
November/December 1997 Vol. 5 Number 3 $4.00US (North America)

the world of 68' micros

Support for Motorola based computer systems and microcontrollers, and the OS-9 operating system

Getting CoCo Disks into (and out of) a PC





TETRIS!

In assembly for the CoCo!

CONTENTS

Editor's Page	*2*
A Letter	*2*
CoCo Emulator Transfer Tricks	*3*
J. Consugar, M. Haaland, R. Cooper	
CoCoFest Vendor Information	*4*
Brian Schubring	
Op Sys Nine : MultiVue	*6*
Rick Ulland	
Tetris!	*9*
Lorne Kelly	
Embedded Programmer	*12*
Paul McKneely	
CoCo 3 Extended Memory : 5	*15*
Herbert Enzman	
A Change of Directory	*18*
Mark Heilpern	
Emulator Sand Patch	*19*
Robert Gault	
New Products from Cloud Nine	*19*
Advertisers Index	*BC*

Don't forget to make plans for the Chicago CoCoFest!!!

POSTMASTER:
If undeliverable return to:
FARNA Systems PB
Box 321
WR, GA 31099

If your address is incorrect, send me a postcard!

The Editor's Page

Sorry, but I was a little rushed getting this issue out on time. What with the holidays and my other hobby (Ramblers), family, and military career, it is sometimes hard to get everything done!

Right after Christmas is always a "dry" spell for everyone. Recovering from the holidays is tough! And the end of this year has been particularly trying for me. But there is light at the end of the tunnel! My "new" Rambler is coming together (1963 Classic wagon with a 1989 Jeep 4.0L fuel injected six) and I'm finally getting this issue out!

You'll notice no letter page this issue. That's the "dry spell" I was referring to. Oh. I got plenty of renewal notes with "thanks" and "keep it up", but rather than print those I thought I'd use the page for more content. And keep the renewals coming in, so I can keep the magazines flowing out!

I've always told you I'd keep you posted on anything that might affect the magazine, and I have some bad news. I won't be able to make the Chicago CoCoFest this year due to military commitments. It was bound to happen sometime or other!

My unit has been tasked to go to the middle east and move an encampment. No, nothing is going on over there! The USAF just decided to combine two smaller troop encampments into one larger, but easier to manage and secure, one. My unit hasn't been anywhere recently, so we got the duty. There will be two 45 day deployments beginning about 15 February. I'll be going on the second, which would begin about the first of April. That is IF I go. There will be 50 people in each deployment, and there are about 200 total in my unit. So some will be staying.

Officially, I have volunteered for the second deployment (my wife has surgery scheduled for early February and will need help around the house for a few weeks), as I haven't been on a lengthy deployment in quite a while and it isn't fair to the other unit members. But I don't make the final decision as to who goes and who doesn't, so there is a possibility I won't be going. Even if I don't go, I won't be able to take leave for the Chicago CoCoFest as we'll be short handed at home. And just in case someone wonders, this isn't a classified mission (VERY routine!) so I'm not divulging anything that isn't a matter of public record. It'll be covered in the base paper when we leave!

I'm going to try to build the next issue a little earlier than usual because of the deployment dates. I won't know for sure if I'm going (I do know I'm not on the first deployment) for a week or two at the earliest. If I don't get everything early enough, the next issue may have to wait until I get back, which should be mid-May. Of course, if the work isn't completed, we may have to stay a few days (or weeks!) more.

If I go, I'll be checking my e-mail when I return. My wife will be collecting all other FARNA mail and holding it for me, so nothing will be missed. Orders and renewals will be processed when I return. Sorry for the inconvenience, but someone has to do their part to protect this great country and its interests!

Thanks for all the support and understanding! I'll keep you all posted.

A Letter! AT306 Tips

1. 68micros: Great that it keeps coming!

2. Someplace in the last issue is a remark that CoCo can fill screen faster than it can be read. A project on my to do list is a "read" cmd substitute for "list" which scrolls a scan line or two at a time so a file can be read while slow scrolling and can be stopped or run up or down.

3. ved stumbles on my AT306 because the cursor keys don't work. ENV/ved_env.file should have some lines edited to be:

```
\kd=$1B,$5B,$42  down
\kl=$1B,$5B,$44  left
\kr=$1B,$5B,$43  right
\ku=$1B,$5B,$41  up
\pd=$1B,$36,$7E  pgdn
\pu=$1B,$35,$7E  pgup
```

4. The AT306 and OS9 v3 didn't have Basic or Runb when purchased. Runb from my MM/1 doesn't run on the AT306/OS9v3. Basic I code programs respond with ":cannot execute". I would like more explanation than that. I notice that the BlackHawk ad lists MWbasic with their MM1b/AT306. Omni Basic looks interesting. But why is there not a Basic09 or MWbasic?

5. Installing OS9 C v3 with the built-in "install" on the distribution disk will clobber the termcap file which umacs wants. So one must grab a termcap from the AT306 OS9 distribution disk (or protect/hide it while "install" is doing its thing).

6. CoCoFest 98!! Good news!!

Fran Walters
72130.3067@compuserve.com

All MWBasic (which is Basic09 for the 68K version of OS9) requires a RunB module. I'm not certain that BlackHawk has ported RunB to the AT306. Even if they have, it may run under OSK v2.4 only, and not v3. Carl didn't include RunB or MW Basic because of the additional licensing fees.

Thanks for the tips! I'm sure other AT306 users will find them very useful.

the world of 68' micros

Publisher:
FARNA Systems PB
P.O. Box 321
Warner Robins, GA 31099-0321

Editor:
Francis (Frank) G. Swygert

Subscriptions:
US/Mexico: $24 per year
Canada: $30 per year
Overseas: $50 per year (airmail)
Back and single issues are cover price.
Overseas add $3.00 one issue, $5.00 two or more for airmail delivery.

The publisher is available via e-mail
dsrtfox@delphi.com

Advertising Rates:
Contact publisher. We have scales to suit every type of business. Special rates for entrepreneurs and "cottage" businesses.

Contributions:
All contributions welcome. Submission constitutes warranty on part of the author that the work is original unless otherwise specified. Publisher reserves the right to edit or reject material without explanation. Editing will be limited to corrections and fitting available space. Authors retain copyright. Submission gives publisher first publication rights and right to reprint in any form with credit given author.

General Information:
Current publication frequency is bimonthly. Frequency and prices subject to change without notice. All opinions expressed herein are those of the individual authors, not necessarily of the publisher. No warranty as to the suitability or operation of any software or hardware modifications is given nor implied under any circumstances. Use of any information in this publication is entirely at the discretion and responsibility of the reader.

All trademarks/names property of their respective owners

ENTIRE CONTENTS COPYRIGHT 1997, FARNA Systems

CoCo EmulatorTransfer Tricks

Porting Double Sided Disks — Joseph Consugar

Despite the fact that I have used all sorts of computers over the years, my favorite is still my CoCo3 and OS9 Level 2. In fact, I bought the CoCo3 emulator specifically so I could run OS9 on my PC.

It took a while, but with the help of Walter Grossman's article in the November/December 1997 issue of "The World of 68' Micros", I was able to create a boot disk for the emulator and get OS9 to come up. Now I was left with one problem; how to transfer all of my OS9 disks to the PC so the emulator could read them.

The emulator comes with the RETRIEVE program to create disk images that can be read by the emulator, but it is designed to work only with single-sided disks. My OS9 disks are a mixture of 5.25" and 3.5" disks, all double-sided.

Transferring the 5.25" disks didn't present a problem. I simply formatted a 3.5" disk as single-sided, copied the files from the 5.25" double-sided disk to the 3.5" single-sided disk, and used the RETRIEVE program to move the disk to the PC. Discovering how to transfer the 3.5" disks without having to do multiple copies was more of a challenge.

Transferring 3.5" Disks

Transferring a complete 3.5" double-sided disk so it can be used with the emulator requires two pieces of software; the OS9 file transfer utility available on the CoCo Files web page (http://people.delphi.com/phxken/COCOFILE.HTML) and a disk editor, like dEd. You must also have an OS9 drive descriptor set up to handle the format of the disk you are trying to transfer (in my particular case, this meant double-sided with 80 tracks).

(**Editor:** *Just any OS9 to MSDOS transfer utility won't work. The one mentioned was specially written to create a virtual disk image on a DOS computer that is compatible with the emulator. If you don't have Web access send a $5 handling fee to FARNA Systems requesting the file. Don't forget to mention the article title and issue date.*)

To perform the transfer, download and install the OS9 file transfer utility on your PC. Go to DOS (or a Windows DOS screen), place the 3.5" disk in the PC's drive, and type the command "OS9 a: -m os9disk.dsk" (this command assumes you have placed the disk in the PC's "a:" drive. If not, change the drive designator to the one approprite for your PC). The result will be an emulator disk image file named os9disk.dsk in your current directory.

Start the emulator and boot OS9. Press F2 to start the virtual drive utility and place the disk image you just created (os9disk.dsk) into the slot corresponding to the drive descriptor you have to handle the disk format. Press <ESC> to return to OS9.

Start the disk editor with a filename corresponding to the disk you just entered in the virtual disk drive utility (e.g., if you placed the disk in slot 1 and are using dEd, the command would be "dEd /d1@" where the "@" symbol means you wish to edit the disk as if it was one big file). Move to LSN0 of the disk and change the value in location $10 from $03 to $02 (or from $07 to $06). Write the sector back to the disk and quit the disk editor. You should now be able to access the disk.

continued on page 17

DOS to Emulator Transfer — Mike Haaland

To tranfer an MS-DOS file to a CoCo emulator virtual disk (.DSK) file you need to create a virtual disk by:

1. When you're in the emulator, click the right mouse button to bring up the main menu. Click on the "L. Virtual Disk Menu" option. This brings up a list of PC drives on the right and the the CoCo disk drives in the upper left.
2. Click on a CoCo drive slot under Diskname. This turns on the cursor in the slot. Now you can type in any name you want to create a disk. If you want to mount an existing disk, click on the down arrow in the upper right to see a list of .DSK file in the current directory and click on the one you want to mount.
3. If a new virtual disk was created, it must be formatted with DSKINI under DECB once it's mounted.
4. Now to transfer to/from a CoCo .DSK image, click the right mouse button to take the emulator to the main menu. Select "P. File Port Utility".
5. This bring up the import/export screen. On the right there are two long boxes. The first one is a listing of the CoCo directory and the second is the list of PC drives.
6. Click the down arrow in the MS-DOS Files window and select the disk image (.DSK) you want to use as the emulator disk to transfer to/from. The contents should be displayed in the CoCo Files box.
7. Now you can move files from MS-DOS to the emulator disk by clicking on the MS-DOS file to transfer in the MS-DOS File box. Or transfer from the emulator disk to the current MS-DOS directory by clicking on a file in the CoCo box.
8. You can set the transfer options on the left by clicking on any bright light blue option.

Emulator/"Real" CoCo Transfers — Rick Cooper

Files are transferred by using the programs named DSKINI.EXE and RETRIEVE.EXE. These programs are included in the CoCo 2 and CoCo 3 emulator packages.

You must have a 5 1/4" drive on your PC, preferably a hi-density drive. I've not had any success using low-density drives.

To move a "real" CoCo disk to the PC:

1. Place the disk in your PC drive (example drive B:).
2. From the DOS prompt in the emulator directory type:
   RETRIEVE B: DISKNAME <ENTER>

Please note the space between the B: and the diskname; it must be there. The diskname is 8 characters long. Don't put an extension on the name (The extension ".DSK" will be assumed and automatically added).

To move an emulator .DSK image to a "real" CoCo 5 1/4" disk:

1. Place a disk in your PC drive.
2. From the DOS prompt in the emulator directory type:
   DSKINI B: DISKNAME <ENTER>

Please note the space between the B: and the diskname. The diskname will be the name of the PC".DSK" file that you are moving to the "real" CoCo disk.

Chicago CoCoFest 98!

THE GLENSIDE COLOR COMPUTER CLUB OF ILLINOIS PRESENTS
THE SEVENTH ANNUAL "LAST" CHICAGO COCOFEST
April 18th & 19th, 1998 (Sat. 10am-5pm; Sun. 10am-3:30pm)

Elgin Holiday Inn

(A Holidome Indoor Recreation Center)
345 W. River Road
Right off intersection of I-90 & IL-31, Same location as past years!
Overnight room rate: $65.00 (plus 10% tax)

Call 1-847-695-5000 for reservations. Be sure to ask for the "Glenside" or "CoCoFEST!" rate. There is a limited number of rooms set aside for the CoCoFest. *These rooms will be released on March 31. They will not be available at the 'Fest rate after that date,* ***so make your reservations early!***

General Admission: $10.00, whole show (Children 10 and under are free)

For further information, general or exhibitor, contact:

Tony Podraza, VP, Spcl Evnts, GCCCI
847-428-3576, VOICE
847-428-0436, BBS
Tonypodraza@juno.com

Mike Knudsen, President, GCCCI
630-665-1394, VOICE
Mknudsen@lucent.com

Brian Schubring, Ast., Fest Coordinator, GCCCI
E-MAil theschu2@juno.com OR theschu3@aol.com

CoCoFEAST!

That's right a CoCo FAMILY DINNER at the HoliDay Inn. Why? So You don't have to drive 'here or there' or try to decide which group you want to spend time with. We can be all together to enjoy the food, and best of all, each others company. There my be a Keynote speaker present. This is planned for Saturday Night about 6:00pm. We need is a MINIMUM of 50 people to reserve a dining room. The tickets will only be available in advance, AT THIS TIME! People to conntact are listed below. The cost will be only $15 U.S. PER Person. We will be able to take paid reservations only up to March 28th, 1998. Please contact one of us for further details.

NOTE: THE CLUB IS NOT MAKING ANY MONEY FOR THE DINNER, NOR PLANS TO. THIS FUNCTION IS ONLY TO PROMOTE A COCO FAMILY GATHERING AT ONE GREAT LOCATION... THE COCOFEST!

If by the specified date we are lacking the number of attendees required, the dinner will be scrapped and a refund will be issued at the Fest.

Afterwards there may be a Musical Monk'O Rama Jam-Session like we had last year with Brother Jeramy, Allen Huffman, and anyone else who wants to bring and instrument and join in!

See Ya'll there in '98!!!

ADDITIONAL COCOFEST VENDOR INFORMATION

The Glenside Color Computer Club of Illinois is a not-for-profit computer club established to assist its members in the learning and better understanding of Tandy's Color Computer. In the pursuit of that goal, we have been priveleged to host, under the auspices of Falsoft Publishing, five Chicago Rainbowfests and one CoCoFEST! sponsored by CoCoPro!. Last year, we organized AND sponsored the Sixth Annual "Last" Chicago CoCoFEST, which was in the Chicagoland area.

This year, Glenside will again sponsor the show as noted above. We would like to continue our tradition of giving door prizes, but we need your help to do so. All vendors are asked to donate items to be distributed through a drawing of admission pass stubs. If you wish your donation to be part of the Grand Prize given on Sunday, please state so. We are also open to donations for the auction held during the show. Proceeds go to Glenside to help offset costs for this and future CoCoFEST's.

Donations MUST be received by April 1, 1998, in order that all prizes and vendors can be properly credited. You will receive a receipt for your donations

EXHIBITOR INFORMATION:

Tables: $35 each (excludes vendor passes)

Full booth exhibits shall be no less than 8'x8'. including one 6' table (draped and skirted), booth sign bearing Exhibitor's company name and booth number, two chairs, and one electrical outlet. Other exhibition needs can be provided at extra cost. Booths may not be shared without the knowledge and written consent of the FEST! Coordinator (inquire!).

Vendor Passes: $5.00 each (max of 2 passes per table)

REGISTRATION MUST TAKE PLACE WITHIN THE MONTHS OF JANUARY AND FEBRUARY, 1998! The FEST! Coordinator MUST have your deposit of $25 by February 28, 1998. The remainder is due by April 1, 1998. Payments received AFTER April 1 will be subject to a 20% LATE FEE based on the outstanding balance. Deposits are NON-REFUNDABLE after April 1, 1998. Send a check for the deposit made payable to "Glenside Color Computer Club" to:

CoCoFest! Glenside Color Computer Club
c/o Tony Podraza, FEST! Coordinator
119 Adobe Circle
Carpentersville, IL 60110-1101

A registration packet will be returned to you. You may, of course, write for the packet first, but the deposit MUST be received by the FEST! Coordinator no later than February 28,1998.

Liabilities and Restrictions:

Sponsor reserves the right to change exhibit hours or cancel the exhibition of its own volition, in which case all deposits and prepayments will be refunded. Should the exhibit be cancelled due to circumstances beyond the exhibitors control, deposits and payments may not be refunded. Neither the sponsors nor its representatives shall be liable for any injuries, loss, or damages in any form. Exhibitor is at all times responsible for its own goods and materials regardless of location.

FARNA Systems

Your most complete source for Color Computer and OS-9 information!

Post Office Box 321
Warner Robins, GA 31099
Phone: 912-328-7859
E-mail: dsrtfox@delphi.com

ADD $3 S&H, $4 CANADA, $10 OVERSEAS

BOOKS:

Mastering OS-9 - $30.00
Completely steps one through learning all aspects of OS-9 on the Color Computer. Easy to follow instructions and tutorials. With a disk full of added utilities and software!

Tandy's Little Wonder - $25.00
History, tech info, hacks, schematics, repairs,... almost EVERYTHING available for the Color Computer! A MUST HAVE for ALL CoCo aficionados, both new and old!!! This is an invaluable resource for those trying to keep the CoCo alive or get back into using it.

Quick Reference Guides
Handy little books contain the most referenced info in easy to find format. Size makes them unobtrusive on your desk. Command syntax, error codes, system calls, etc.
CoCo OS-9 Level II : $5.00
OS-9/68000 : $7.00

Complete Disto Schematic set: $15
Complete set of all Disto product schematics. Great to have... needed for repairs!

SOFTWARE:

CoCo Family Recorder: Best genealogy record keeper EVER for the CoCo! Requires CoCo3, two drives (40 track for OS-9) and 80 cols.
DECB: $15.00 OS-9: $20.00

DigiTech Pro: $10.00
Add sounds to your BASIC and M/L programs! Very easy to use. User must make simple cable for sound input through joystick port. Requires CoCo3, DECB, 512K.

ADOS: Best ever enhancement for DECB! Double sided drives, 40/80 tracks, fast formats, extra and enhanced commands!
Original (CoCo 1/2/3) **: $10.00**
ADOS 3 (CoCo 3 only) **: $20.00**
Extended ADOS 3 (CoCo 3 only, requires ADOS 3, support for 512K-2MB, RAM drives, 40/80 track drives mixed) **: $30.00**
ADOS 3/EADOS 3 Combo: $40.00

Pixel Blaster - $12.00
High speed graphics tools for CoCo 3 OS-9 Level II. Easily speed up performance of your graphics programs! Designed especially for game programmers!

Patch OS-9 - $7.00
Latest versions of all popular utils and new commands with complete documentation. Auto-installer requires 2 40T DS drives (one may be larger).

TuneUp : $20.00
Don't have a 6309? You can still take advantage of Nitro software technology! Many OS-9 Level II modules rewritten for improved speed with the stock 6809!

Thexder OS-9
Shanghai OS-9 : $25.00 each
Transfer your ROM Pack game code to an OS-9 disk! Please send manual or ROM Pack to verify ownership of original.

Rusty : $20.00
Launch DECB programs from OS-9! Load DECB programs from OS-9 hard drive!

NitrOS-9:

Nitro speeds up OS-9 from 20-50% depending on the system calls used. This is accomplished by completely rewriting OS-9 to use all the added features of the Hitachi 6309 processor. Many routines were streamlined on top of the added functions! The fastest thing for the CoCo3! Easy install script! 6309 required.
Level 3 adds even more versatility to Nitro! RBF and SCF file managers are given separate blocks of memory then switched in and out as needed. Adds 16K to system RAM... great for adding many devices!
NitrOS-9 V.2.0: $35.00
NitrOS-9 Level 3: $20.00
SAVE $10! V.2.0 & Level 3: $45.00

The AT306 OS-9 Single Board Computer

AT306 Motherboard Specs:

16 bit PC/AT I/O Bus (three slots)
MC68306 CPU at 16.67MHz
Four 30 Pin SIMM Sockets
IDE Hard Drive Interface
Floppy Drive Interface (180K-2.88M)
Two 16 byte Fast Serial Ports (up to 115K baud)
Two "Terminal" Serial Ports (no modem)
Bidirectional Parallel Port
Real-time clock
PC/AT Keyboard Controller (five pin DIN)

Included Software Package:

"Personal" OS-9/68000 Vr 3.0
(Industrial with RBF)
MGR Graphical Windowing Environment
with full documentation
Drivers for Tseng W32i
and Trident 8900 VGA cards
Drivers for Future Domain 1680
and Adaptec AAH15xx SCSI cards
Many PD and customized utilities and tools

The AT306 is a fully integrated single board computer. It is designed to use standard PC/AT type components. Sized the same as a "Baby AT" board (approximately 8" square). Compact and inexpensive enough to be used as an embedded controller! Use with a terminal (or terminal emulation software on another computer) or with a video card as a console system. Basic OS-9 drivers are in ROM, making the system easy to get started with.

HACKERS MINI KIT (FARNA-11100): Includes AT306 board, OS-9 and drivers, util software, assembly instructions/tips, T8900 1MB video card. Add your own case, keyboard, drives, and monitor! *ONLY $500!*

Call for a quote on turn-key systems and quantity pricing.
Warranty is 90 days for labor & setup, components limited to manufacturers warranty.

Microware Programmers Package -
Licensed copies of Microware C compiler, Assembler, Debugger, and many other tools!
***With system purchase:* $65.00 *Without system:* $85.00**

operating system nine

Rick Ulland

MultiView!

North & South of Computing

Computer magazines talk alot about applications. This is fine ... applications are what you buy a computer for. Right? Well ... it does make a great way to sell software but it's important to look at why computers have historically been used this way.

There are two directions an operating system can take. An obvious solution with limited hardware is to devote 100% of the machines resources to the job at hand. Made lots of sense for early home computers. By now we've seen both pretty and elegant versions and the concept 'feels' familiar – to do a job, you buy a tool (program) and follow the directions. You buy enough tools, they upgrade 'em next year. The only problem is your tool had better have all the features needed because if it can't do it, it can't easily be done. At best, this makes for some big programs.

There is a second way to use computers. Here, the data set is king. Instead of loading Quicken (tm,r,@,etc, etc) you go to the directory with the bankbook data. The difference isn't obvious when you've only got one app anyway, but eventually the ol' bankbook directory will have dozens of appletts sharing facts.

Folks who do mainframes and minis are comfortable here, but this scheme never caught the mainstream. It takes a slightly different sort of computer. Instead of opening a known hardware platform and just lying there, the opsys has to steal a little of the machines power and manage it's users. From a privileged vantage, it keeps multiple applets happy with a single disk drive and monitor without locking users in some tiny utility. But the management costs are high, and single-use computers quickly allowed some pretty advanced looking tools. With a new dll and wholly owned hardware, you can do some mean byte mashing! This only reinforces the tool per job mentality and the glitz hides the fact a single task opsys needs an 'Office Suite' to handle the scheduling.

It also needs a blazing fast CPU to get all these sequential operations out of the way fast enough to keep the GUI running! Speedometers are a popular diversion. Paradoxically, the single program machine usually appears to be getting more done -- once it moves, it really moves, zapping out a screen before hiding behind the hourglass to catch up computing during the user's wow time.

Trying to emulate this console-centric view breeds big, bloated windowing packages that allow workstations to divide and conquer any conceivable cpu upgrade (evenly among all users). It's no wonder few considered such a scheme practical on a desktop!

Enter MultiVue

By overlaying a GUI on OS9's unix like command structure, MultiVue (MVue) provides our collection of tiny appletts with a modern interface. With OS9 as a base, the GUI doesn't have to provide multiprocess scheduling – it's only job is to replace the text screen and keyboard interface. If the CoCo hadn't died out from under it, WindInt coulda beena contenda.

By creating and maintaining clickable menus at the opsys level it brings real time to the user interface. With applications relieved of the need to talk to a human, they can concentrate on the job at hand and OS9's inherent strengths keep our motley collection of software chugging along without the burps and hiccups common in wintel. You can even close and reopen programs!

In answer to the recent CoCo-list question 'What does real time response provide the average CoCo user?' I point to a working cursor (someplace) at all times. Remember, these applications were not specially written to appear to multitask! In fact, if you learn to roll over the delays with extra windows a MultiVue CoCo can run with a 386 Windows box. This is not the same as saying it's as fast as a 386 -- screen resolution is less and the files much smaller. It's the tortoise and hare -- does your bank book really need more than 10 decimal palces?

Those that have built a real MultiVue system know that at first, speed is not an issue. Just getting a usable system up is enough. To begin, here aren't many programs for WindInt. Other than Hypertech (ShellMate, MVCanvas) and Gravity Studios (Planet Engine) none come to mind.

Even so, a stock CoCo staggers badly if you throw too much OS9 at it. Tinkering types can patch the CoCo to a reasonable level – get a hard disk to avoid the halting floppy, patch to Ed9 clock for rs232 work, and any serious terminals need buffered serial like Fast232 (CoNect). With more than one port, tie irq (CART) on the mpi ("strap" the CART lines). There are still limits imposed by the hardware, mainly visible as a periodic spasm/crash. Those little keyscan and mousecount routines still manage to get in the way sometimes.

If you don't write programs, WindInt remains just an interesting factoid so we'll stick to the highlight reel. Some of the 'additions' are mere gingerbread – shadowed dialog boxes are pretty but a simple change of background color would serve and not cost a gfx console.

The menu windows are the meat of the package. A few simple system calls can arrange the number and contents of menus on the bar, and the mouse handler takes care of monitoring them. The mouse will signal when clicked, then report the scaled X/Y data range or return menu and selection numbers if outside the work area.

Scrollbars are simply another menu (actually id# 4-7 for 4 ways), and a hotkey press is returned as a selection from menu 8. Stock menus Tandy(20), File(21) and Edit(22) are predefined for any program, which can add custom menus 128-255 to total 10 menus of 20 items each. This is great news if you've got a data cruncher and need an interface. Create your own menu library with a few includes.

For the rest of us, the main thing MultiVue supplies is gshell. Download Mark Marlette's excellent patch set and you've got almost everything you need to make the CoCo easy to use next time. When it's time to do email, I want to click the phone and go. MultiVue supplies that, at the cost of 'hacking CoCo' occasionally. If you don't think it's sort of fun to draw new icons and invert procedure files, and you don't have an other who does, it will take some patience to replacing the pretty interface supplied with a 'suite'.

All we're given is the pathlist. This is literally the directory structure of a system's hard drive but can also be read as a series of menu choices. Consider the path /user/desktop/graphics/images ... this is a logical set of menu choices when looking for a big pile of images, and thats what you'll find after clicking a few file folder icons. The programs that work with them are nowhere to be seen, which makes for a nice clean directory listing but leaves the user a little lost. This is normal OS9 but even with a printed list /

command line various on an obscure contraction of a process who's name is long forgotten is not all that useful!

Gshell adds small data files to link these programs to the directory (by location), and the specific data file type (by file extension). The directory screen starts with the AIF list followed by the files themselves. Works like a win toolbar, just click up top to load and run as many bits of the graphics suite as needed, or launch a data file directly. Imagine this interface projected to the animated audio icon level and you wonder why "They" stretched so far for 'weblike'?

Giving it a go

It seems lots of folks stumble getting MultiVue installed. It does require a change to the operating system itself. What's actually not that difficult a process is hidden inside Tandy's auto installer, which promptly explodes if you throw anything interesting at it. All it's doing is replacing the standard Lvl2 Grflnt module with Windlnt and doubling the number of installed window descriptors for some GUI room. The rest of the install is simply copy the program with the only wrinkle adding a /dd/CMDS/ICON directory.

If your hacking has been limited to modpatch and cobbler, there's some prep work undone. You could build a Tandy MVue and simply repatch everything, but this will come up again. Better to save any changes made to your pre-multivue system as discrete modules. This way, OS9's system builder of last resort (OS9Gen) can recreate any version from a selection of bootlist files.

Luckily, you've just gotten the tool to do this. MultiVue includes the OS9 save utility, hidden away in the pmpts file. Load pmpts off the MVue disk and let save save itself (command: save save save). Now you can save anything you've modpatched as an official system module. Common changes will be to descriptors (/term, /t2, /dd, etc) although almost every OS9 module has been patched by somebody. Save stores things in the execution directory (usually you are saving executable program modules) so you'll want to divert that with a chx to wherever you keep your system modules, saving each update under a descriptive name like d1_80d6ms.dd. Copy all the MVue supplied modules in as well.

You'll find both the stock and MultiVue disks include a bootlist file. Comparing the two, changes are limited to replacing Grflnt with Windlnt, and adding a half dozen window descriptors. You'll want to edit in your updates, add hard disk drivers and so on.

Once the list is complete OS9Gen makes short work of a new bare boot. On a hard disk system, this is close to the end. Copy cc3go and shell to make a bootable floppy and make sure all MVue's programs have been moved to the HD. You can decide if you want MVue to autostart – what they've done is copy multistart to a file with the key name autoex. Rename autoex MVue, and type that to manually start the program.

Even if you don't always run MultiVue, you might want to take advantage of its

From: *Dennis Bathory-Kitsz*

Hi folks! I've been hiding out in Vermont, but since it's the 10th anniversary of my company Green Mountain Micro's demise, I thought it might be time to put in an appearance here.

About 150 copies of 'Learning the 6809' (book only) remain, which I'd be happy to offer at $10 postpaid to anyone interested. If at least 10 people also want the original tapes, I'd be pleased to make up a set of those as well.

One of these days I'll tell my own tale ... amusing indeed...

Dennis Bathory-Kitsz
RD 2 Box 2770
Cox Brook Road
Northfield, Vermont 05663

<bathory@maltedmedia.com>
Malted/Media:
http://www.maltedmedia.com/

NEW PRODUCT ANNOUNCEMENT:

Nickolas Marentes is proud to release

** PAC-MAN **

"A tribute to the great game"

For the Tandy Color Computer 3 with 512K RAM and Disk Drive

Finally! A version of the 1980 classic that is so similar to the original that you will think you ARE playing the original. Many of the original's features and characteristics have been included to make this game as faithful to the original Namco classic as possible. Fun, clean, violence free, 80's style entertainment for the whole family.

Features include:

* Most of the original sound effects
* Accurate replica of the original maze
* Accurate display of graphics and animations
* Many of the originals game play elements
* Coded in 100 percent 6809 assembly language
* Runs at 60 fps with 2 channel digital sound
* Keyboard and Joystick controls
* Reduced function DEMO version available as Freeware.
* Low price for full registered version ($20)

Get the best version of this historic game for your CoCo3 today!

Available from:

- USA -
Rick's Computer Enterprises, P.O. Box 276, Liberty, KY 42539
Internet Page: www.voicenet.com/~swyss/cfdm.html
E-mail: rcooper@kih.net

- AUSTRALIA -
Nickolas Marentes, P.O. Box 2003, Runcorn, 4113, QLD.
Internet Page: www.launch.net.au/~stauros/nickpage/ (FREE DEMO!!)
E-mail: N.Marentes@mailbox.uq.edu.au

Pac-man is the registered trademark and property of Namco/Midway.
Money collected is payment for the work involved in the development of the 6809 code.
The author has not seen or copied any of the original's Z-80 code.

control applet, which sets system colors, key speed and the like. To invoke all MVue settings in any boot just add the line control -e to the startup file. To change system settings, fire MVue and run control ... macho types can edit /dd/sys/env.file, a Gatezian ini file that explains itself.

Floppy users have a small problem to work around. What MVue wants is constant access to all executables in /dd/CMDS, all icons in /dd/CMDS/ICONS, and all system data in /dd/SYS. This is very hard to do when /dd is a 360K floppy. The best luck I've had is with a 2 disk boot, where startup loads common utilities from 'merged files' then a second disk is swapped in for normal running. A 360K floppy can add 200K of 'RAMdisk' in preloaded commands, leaving 200K RAM to run them in. The second disk always has the complete ICONS dir for all data disks, and the complete SYS. The little bit of leftover room is filled with commands not preloaded, and drive #2 can combine specific programs with their data – for example the non-boot contents of a Tandy program disk.

I've heard of a trick here that I've never had identical sized floppies to try. The idea is to make up a boot disk compete with CMDS, CMDS/ICONS, and SYS directories, then use backup to make identical copies of this disk. Now make the replacement /d0 disks by deleting and adding files to one of the boot disk clones. You'll end up with multiple program disks that don't leave out any important files, since each has to be explicitly deleted. The good part is this leaves the CMDS dirs at identical locations on each former clone. You don't have to chx when they're swapped! With MVue's change execute dir hidden off in the menus, this is no small favor.

Once MVue boots, the process of reorganizing begins. If you've been following standard OS9 practice, the placement work is mainly done, perhaps a little renaming to emphasize the menu nature of the directory tree. The filenames may be done also.

A typical program uses a three letter extension to identify its datafiles and MVue sorts data files into categories by the same extensions. Any file named AIF.xxx is assumed to contain info to launch the appropriate program. The fine point is these AIF links aren't part of some system wide database, but loaded on the fly from the data directories encountered as the user climbs among the tree.

Where you put an aif is as important as whats in it. For speed, you want the user to pick up often used AIFs early so later directories don't need a separate copy. Little used ones can be placed far up the tree where they will rarely be loaded. You can even control the programs offered by the system, again using the pathlist.

When an ICON is loaded, it over rides any future aifs using that extension. If icons for system administration utilities are in SYS and the user goes there, he'll carry the 'system administrator' aifs with him until flushing the cache by selecting a drive icon.

A typical example – the /desktop/write path assigns .doc to dynastar, a word processor. The /sys/docs path links .doc to vu, for fast file listing. To edit one of the /sys/doc files user climbs to the desktop (locking in DynaStar), then climbs over to docs without using the /h0 button. The vu aif.doc can't load, so all the doc files have word processor icons. A parting bit of placement trivia – name each one in capitals. A gsort will put AIF.XXX ahead of almost anything.

Programming

An AIF is a simple creature, nine lines of text representing program name, command line options, path to icon, ram size (0 for default), screen type, x size, y size, forecolor, and backcolor. Any text editor can make them, but you'll need to download an icon editor to create the matching pictures. For examples, we'll list a few standard AIFs that should have been supplied with MultiVue, but weren't.

First, an 80 column text shell or basic09 workspace. Add separate AIFs for a couple of gfx screen versions:

shell	basic09
(blank line)	#32K
icons/icon.ops	icons/icon.b09
0	128
7	7
80	32
24	16
1	1
0	0

After the command line, notice the shell aifs blank line adds no command line options while the basic09 version calls for a 32K workspace . A third alternative (after patching gshell) is to put a question mark here. The machine will then ask the user for command line options at run time. You have to know how to type them in so a final option is multiple AIFs with commonly used option settings.

Note that both examples use a type 7 text screen, 80x24 initial size. If you think about it, the 2 lines following seem redundant – the system should know that a type 7 screen is an 80x24 box. What they do is allow sizeable windows. The example uses a 32x16 shell to match the older Basic09 screen format. Simply specify a window size smaller than full screen and clicking causes a place screen to appear. The user can position this minimum size window among other running programs, or drag it up to full window size.

Then there are packed Basic09 programs. The normal method of launching (runb program) works, but translates to first line (programname)=runb, second line (options)= B09 program name. This leaves all the B09 icons titled 'runb'. While the gshell patch removes the need to explicitly call runb you have to manually edit all the AIFs to take advantage, moving line 2 to line 1 to get meaningful AIF titles.

Another early project should be an AIF for the control program. It already has a menu slot, but the menu selection uses the exsisting (usually 4 color) screen. Setup the AIF to fire control (no options) in a 16 color window and set ALL the colors with the mouse! You can also make an AIF for multistart, and click for a second, separate gshell.

For years, I thought this couldn't be done – attempting to fire two gshells in startup leaves one with a dead menu bar. The aif sometimes creates a menu bar that appears dead but a click reveals things are working fine.

And I'm out of time! As usual, we have a disk for those without modem – $5 S&H, specify 158 (single sided 5.25"), 360 (double sided 5.25"), or 720K (3.5") format. More later.

CoNect

1629 South 61st Street
West Allis, WI 53214
(pulland@omnifest.uwm.edu)
414-328-4043

Tetris!

An assembly game for the CoCo 3

Lorne Kelly

This is an assembly language version of Tetris that I wrote. Run The basic program to generate DATA.BIN, assemble the assembly (absolute origin), load data.bin, and start execution at /SS. I know this could be done much better - it was just for fun...

```
00010 **************************
00020 * TETRIS BY LORNE KELLY  *
00030 * 11/23/96 THIS PROGRAM  *
00040 * REQUIRES A BLOCK OF    *
00050 * DATA WITH BYTEMAPS FOR *
00060 * GRAPHICS ETC. (GENERATE*
00070 * WITH TETRIS.BAS)       *
00080 **************************
00100  ORG   $6600
00105 BLANK EQU   $80
                  Background color
00110 SCREEN  EQU   $400
                  Screen starts here
00120 SDATA EQU   $6000
                  Data to draw screen
00130 BLDATA   EQU   $6200
                  Data for block shapes
00140 AGNMSG EQU   $65A0
                  'Play again?' message
00150 STACK EQU   $5FFF
                  User stack goes here
00160 POLCAT   EQU   $A000
                  Scan keyboard
00170
00180 CURBL FCB $2
                  Current block variable
00190 CURRO    FCB $18
                  Current rotation
00200 CURCOR FDB $404
                  Current coordinate
00210
00220 *draw the game screen
00230 CLRSCR  LDX #SCREEN
00240 LOOP1 LDY SDATA-SCREEN,X
00250  STY ,X++
00260  CMPX #$600
00270  BNE LOOP1
00280  RTS
00285
00290 * Erase block at old co-coordi-
          nates, Test new pos, and draw.
00300 DRAWBL  LDX #BLDATA
00310  PSHU  X,Y,D
00320  LDY CURCOR
00330  LDD CURBL
   A=Block(0-7) B=Rotation(0,8,16, or24)
00340  ABX    Rotation
00350  LDB #128
          Size of a block (all 4 rotations)
00360  MUL    Block
00370  LEAX  D,X  Load x with offset
00380  LDB #8    Block is 8 bytes wide
00390 ERASE LDA ,X
00400  CMPA  #BLANK
00410  BEQ SKIP10
00420  LDA #BLANK
00430  STA ,Y
00440 SKIP10 LDA 32,X
00450  CMPA  #BLANK
00460  BEQ SKIP11
00470  LDA #BLANK
00480  STA 32,Y
00490 SKIP11 LDA 64,X
00500  CMPA  #BLANK
00510  BEQ SKIP12
00520  LDA #BLANK
00530  STA 64,Y
00540 SKIP12 LDA 96,X
00550  CMPA  #BLANK
00560  BEQ SKIP13
00570  LDA #BLANK
00580  STA 96,Y
00590 SKIP13 LEAX  1,X
00600  LEAY  1,Y
00610  DECB
00620  BNE ERASE
00630 *Test for room at new position
00640  PULU  D,Y,X
00650 TDNEW   PSHU X,Y,D
00660  ABX
00670  LDB #128
00680  MUL
00690  LEAX  D,X
00700  LDB #8
00710 TEST   LDA ,X
00720  CMPA  #BLANK
00730  BEQ SKIP6
00740  LDA ,Y
00750  CMPA  #BLANK
00760  BNE NOROOM
00770 SKIP6   LDA 32,X
00780  CMPA  #BLANK
00790  BEQ SKIP7
00800  LDA 32,Y
00810  CMPA  #BLANK
00820  BNE NOROOM
00830 SKIP7   LDA 64,X
00840  CMPA  #BLANK
00850  BEQ SKIP8
00860  LDA 64,Y
00870  CMPA  #BLANK
00880  BNE NOROOM
00890 SKIP8   LDA 96,X
00900  CMPA  #BLANK
00910  BEQ SKIP9
00920  LDA 96,Y
00930  CMPA  #BLANK
00940  BNE NOROOM
00950 SKIP9   LEAX  1,X
00960  LEAY  1,Y
00970  DECB
00980  BNE TEST
00990 *TEST SUCCESS! Store new
          pos,block & rotation
01000  PULU  D,Y,X
01010  STY CURCOR
01020  STD CURBL
01030  LDY #0
01040  PSHU  X,Y,D
01050 *Draw at stored location
          & rotation
01060 NOROOM LDY CURCOR
01070  LDX #BLDATA
01080  LDD CURBL
01090  ABX
01100  LDB #128
01110  MUL
01120  LEAX  D,X
01130  LDB #8
01140 DRAW  LDA ,X
01150  CMPA  #BLANK
01160  BEQ SKIP
01170  STA ,Y
01180 SKIP LDA 32,X
01190  CMPA  #BLANK
01200  BEQ SKIP1
01210  STA 32,Y
01220 SKIP1   LDA 64,X
01230  CMPA  #BLANK
01240  BEQ SKIP2
01250  STA 64,Y
01260 SKIP2   LDA 96,X
01270  CMPA  #BLANK
01280  BEQ SKIP3
01290  STA 96,Y
01300 SKIP3   LEAX  1,X
01310  LEAY  1,Y
01320  DECB
01330  BNE DRAW
01340  PULU  D,Y,X
01350  RTS
01355
01360 *EXEC location
01370 SSLDU #STACK
01380  LBSR  CLRSCR
01390  LBSR  NEWBL
01395 *Main Loop Area
01400 LOOP15   JSR [POLCAT]
01410  JSR CHOICE
01415  LDY #0    Y is fail flag
01420  LBSR  FALL
01425  CMPY #0    did fall fail?
01427  LBNE  AGAIN yes! End game.
01430  LBSR  RNDAB
01440  BRA LOOP15
01445
01450 CHOICE   CMPA #0
01460  BEQ NOKEY
01490  CMPA #$4B  K
01500  BEQ ROTATE
01530  CMPA #$4A   J
01540  BEQ MOVEL
01550  CMPA #$4C  L
01560  BEQ MOVER
01570  CMPA #$20 SPAC
01580  LBEQ  DROPBL
01590 NOKEY RTS
01595
01600 MOVELLDY CURCOR
01610  LEAY  -2,Y
01620  LDA CURBL
01630  LDB CURRO
01640  LBSR  DRAWBL
01650  RTS
01655
01660 MOVER   LDY CURCOR
01670  LEAY  2,Y
01680  LDA CURBL
```

```
01690  LDB CURRO
01700  LBSR  DRAWBL
01710  RTS
01705
01720 ROTATE   LDYCURCOR
01730  LDA CURBL
01740  LDB CURRO
01750  ADDB  #8next rotation
01760  CMPB  #$20   last rotation
01770  BNE SKIP5
01780  CLRB      return to 0 rotation
01790 SKIP5  LBSR  DRAWBL
01800  RTS
01803
01805 *Drop block to bottom
01810 DROPBL  LDYCURCOR
01820  LDD CURBL
01830  LEAY  +32,Y  one line
01840  LBSR  DRAWBL
01850  LDX #$750
01860 LOOP16  LEAX  -1,X
01870  BNE LOOP16
01880  CMPY #0
01890  BEQ DROPBL
01900 NEWBL   BSR   TETRIS
01910  BSR RNDAB
01920  STD CURBL
01930  LDY #SCREEN+5
01940  STY CURCOR
01950  LBSR  TDNEW
01980  RTS
01983
01985 *Generate random number
          for block & rotation
01990 RNDPTR  FDB$A000
02000 RSTPTR  LDX#$A000
02010  STX RNDPTR
02020 RNDABLDX RNDPTR
02030  LEAX  1,X
02040  CMPX #$B000
02050  BEQ RSTPTR
02060  STX RNDPTR
02070  LDB #$18
02080  ANDB [RNDPTR]
02090  LDA #$7
02100  ANDA [RNDPTR]
02110  CMPA #$7
02120  BNE SKIP20
02130  LDA #3
02140 SKIP20 RTS
02143
02145 * Check if time for block to fall
02150 FALTMR   FDB$500
02160 FALL LDX FALTMR
02170  LEAX  -1,X
02180  CMPX #0
02190  BEQ FALRST
02200  STX FALTMR
02210  RTS
02220 FALRST   LDX#$500
02230  STX FALTMR
02240  LDY CURCOR
02250  LDD CURBL
02260  LEAY  32,Y
02270  LBSR  DRAWBL
02280  CMPY #0
02290  BNE NEWBL
02300  RTS
02304
02305 *test for full line (a tetris)
02310 TETROW FDBSCREEN+1
02320 TETRISLDY TETROW
02330  LEAY  32,Y
02340  CMPY #$5E1
02350  BEQ TETRST
02360  STY TETROW
02370  LDA #21
02380 LOOP25  LDB,Y+
02390  CMPB #BLANK
02400  BEQ TETRIS
02410  DECA
02420  BNE LOOP25
02430  BRA TETYES
02440 TETRST  LDY#SCREEN+1
02450  STY TETROW
02460  RTS
02465 *clear line and start test over
02470 TETYES  LDY TETROW
02480  LDB #10
02490 LOOP26  LDX-32,Y
02500  STX ,Y++
02510  DECB
02520  BNE LOOP26
02530  LDY TETROW
02540  LEAY  -32,Y
02550  STY TETROW
02560  CMPY #SCREEN+1
02570  BEQ DONE
02580  BRA TETYES
02590 DONE  BRA TETRIS
02593
02595 * Play again?
02600 AGAIN  LDX #AGNMSG
02610  LDY #$4E0
02620 LOOP30  LDD,X++
02630  STD ,Y++
02640  CMPY #$540
02650  BNE LOOP30
02660 LOOP31  JSR [POLCAT]
02670  CMPA #$4E  N
02680  LBEQ  QUIT
02690  CMPA #$59  Y
02700  LBEQ  SS
02710  BRA LOOP31
02720 QUIT RTS Change to swi for zbug
02730  END
```

DATA.BAS

```
10 CLEAR 2000,&H6000
20 FOR X = 1 TO 15
30 PRINT CHR$(131);
40 PRINT STRING$(20,128);CHR$(131);
50 PRINT STRING$(1,128)
60 NEXT X
70 PRINT CHR$(128);
80 FOR X = 1 TO 10
90 PRINT CHR$(132);CHR$(136);
100 NEXT X
110 PRINT STRING$(2,128);
120 PRINT @2*32+24,"TETRIS!"
130 PRINT @3*32+24,"———"
140 PRINT @1*32+24,"———"
150 PRINT @5*32+24,"J LEFT"
160 PRINT @6*32+24,"K TURN"
170 PRINT @7*32+24,"L RIGHT"
180 PRINT @11*32+26,"1.0"
190 PRINT @10*32+24,"VERSION"
200 PRINT @9*32+24,"11/2/96";
210 PRINT @13*32+25,"LORNE"
220 PRINT @14*32+25,"KELLY"
230 ' COPY UP TO $6000
240 DEST=&H6000
250 FOR X = &H400 TO &H5FF
260 POKE DEST,PEEK(X)
270 DEST=DEST+1
280 NEXT X
290 'DRAW BLOCKS PAGE 1
10000 'white square
10010 DATA 207,0,0,0,0,0,0,0,1,1,0,0,1,1,0,0
10020 DATA 207,0,0,0,0,0,0,0,1,1,0,0,1,1,0,0
10030 DATA 207,0,0,0,0,0,0,0,1,1,0,0,1,1,0,0
10040 DATA 207,0,0,0,0,0,0,0,1,1,0,0,1,1,0,0
10050 'red line
10060 DATA 191,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0
10070 DATA 191,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0
10080 DATA 191,0,0,1,0,0,0,1,0,0,0,1,0,0,0,1,0
10090 DATA 191,0,0,0,0,0,0,0,0,1,1,1,1,0,0,0,0
10100 'pale blue T
10110 DATA 223,0,0,0,0,0,0,0,0, 1,1,1,0,0,1,0,0
10120 DATA 223,0,0,0,0,0,1,0,0,1,1,0,0,0,1,0,0
10130 DATA 223,0,0,0,0,0,1,0,0,1,1,1,0,0,0,0,0
10140 DATA 223,0,0,0,0,0,1,0,0,0,1,1,0,0,1,0,0
10150 ' green step
10160 DATA 143,0,0,0,0,0,0,0,0,1,1,0,0,0,1,1,0
10170 DATA 143,0,0,0,0,0,0,0,1,0,0,1,1,0,0,1,0,0
10180 DATA 143,0,0,0,0,0,0,0,0,0,1,1,0,0,0,1,1,0
10190 DATA 143,0,0,0,0,0,0,0,1,0,0,1,1,0,0,1,0,0
10200 READ A
10210 FOR Y = 0 TO 3
10220 FOR X = 0 TO 3
10230 READ C$
10240 IF C$="0" THEN POKE&H400+O+X*2+
(Y*32),128:POKE &H401+O+X*2+(Y*32),128
10250 IF C$="1" THEN POKE&H400+O+X*2+
(Y*32),A:POKE&H401+O+X*2+(Y*32),A
10260 NEXT X,Y
10270 O=O+8:IFO/32=INT(O/32)THENO=O+
32*3
10280 IF O >12*32+30 THEN GOTO 10300
10290 GOTO10200
10300 'REM COPY UP
10310 DEST=&H6200
10320 FOR X = &H400 TO &H5FF
10330 POKE DEST,PEEK(X)
10340 DEST=DEST+1
10350 NEXT X
10360 'DRAW BLOCKS PAGE 2
20000 ' BLUE REV STEP
20010 DATA 175,0,0,0,0,0,0,0,0,1,1,0,1,1,0,0
20020 DATA 175,0,0,0,0,1,0,0,0,1,1,0,0,0,1,0,0
20030 DATA 175,0,0,0,0,0,0,0,0,0,1,1,0,1,1,0,0
20040 DATA 175,0,0,0,0,1,0,0,0,1,1,0,0,0,1,0,0
20050 ' YELLOW L
20060 DATA 159,0,0,0,0,1,0,0,0,1,0,0,0,1,1,0,0
20070 DATA 159,0,0,0,0,0,0,0,1,0,1,1,1,0,0,0,0
20080 DATA 159,0,0,0,0,0,1,1,0,0,0,1,0,0,0,1,0
20090 DATA 159,0,0,0,0,0,0,0,0,1,1,1,0,1,0,0,0
20100 ' PURPLE REV L
20110 DATA 239,0,0,0,0,0,0,1,0, 0,0,1,0,0,1,1,0
20120 DATA 239,0,0,0,0,1,0,0,0,1,1,1,0,0,0,0,0
20130 DATA 239,0,0,0,0,1,1,0,0,1,0,0,0,1,0,0,0
20140 DATA 239,0,0,0,0,1,1,1,0,0,0,1,0,0,0,0,0
20150 'rem blank
20160 DATA 143,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
20170 DATA 143,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
20180 DATA 143,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
20190 DATA 143,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0
20195 O = 0
20200 READ A
20210 FOR Y = 0 TO 3
20220 FOR X = 0 TO 3
20230 READ C$
20240 IF C$="0" THEN POKE &H400+O+X*2+
```

HawkSoft

28456 S.R. 2, New Carlisle, IN 46552
219-654-7080 eves & ends MO, Check, COD; US Funds
Shipping included for US, Canada, & Mexico

MM/1 Products (OS-9/68000)

CDF $50.00 - CD-ROM File Manager! Unlock a wealth of files on CD with the MM/1! Read most text and some graphics from MS-DOS type CDs.

VCDP $50.00 - New Virtual CD Player allows you to play audio CDs on your MM/1! Graphical interface emulates a physical CD player. Requires SCSI interface and NEC CD-ROM drive.

KLOCK $20.00 - Optional Cuckoo on the hour and half hour!! Continuously displays the digital time and date on the /term screen or on all open screens. Requires I/O board, I/O cable, audio cable, and speakers.

WAVES vr 1.5 $30.00 - Now supports 8SVX and WAV files. Allows you to save and play all or any part of a sound file. Merge files or split into pieces. Record, edit, and save files; change playback/record speed. Convert mono to stereo and vice-versa! Record and play requires I/O board, cable, and audio equipment.

MM/1 SOUND CABLE $10.00 - Connects MM/1 sound port to stereo equipment for recording and playback.

GNOP $5.00 - Award winning version of PONG(tm) exclusively for the MM/1. You'll go crazytrying to beat the clock and keep that @#$%& ball in line! Professional pongists everywhere swear by (at) it! Requires MM/1, mouse, and lots of patience.

CoCo Products (DECB)

HOME CONTROL $20.00 - Put your old TRS-80 Color Computer Plug n' Power controller back on the job with your CoCo3! Control up to 256 modules, 99 events! Compatible with X-10 modules.

HI & LO RES JOYSTICK ADAPTER $27.00 - Tandy Hi-Res adapter or no adapter at the flick of a switch! No more plug and unplugging of the joystick!

KEYBOARD CABLE $25.00 - Five foot extender cable for CoCo 2 and 3. Custom lengths available.

MYDOS $15.00 - Customizable, EPROMable DECB enhancement. The commands and options Tandy left out! Supports double sided and 40 track drives, 6ms disk access, set CMP or RGB palettes on power-up, come up in any screen size, Speech and Sound Cartridge support, point and click mouse directory, and MORE OPTIONS than you can shake a stick at! Requires CoCo3 and DECB 2.1.

DOMINATION $18.00 - Multi-Player strategy game. Battle other players armies to take control of the planet. Play on a hi-res map. Become a Planet-Lord today! Requires CoCo3, disk drive, and joystick or mouse.

SMALL GRAFX ETC.

"Y" and "TRI" cables. Special 40 pin male/female end connectors, priced EACH CONNECTOR -		$6.50
Rainbow 40 wire ribbon cable, per foot -		$1.00
Hitachi 63B09E CPU and socket -		$13.00
MPI Upgrades for all small MPIs (satellite board) -		$10.00
Serial to Parallel Convertor with 64K buffer and external power supply -	**NOW ONLY**	**$28.00!!!**
Serial to Parallel Convertor (no buffer) and external power supply -	**ONLY**	**$18.00!!!**
2400 baud Hayes compatible external modems -		$15.00
Serial to Parallel Convertor or Modem cable (4 pin to 25 pin) -		$5.00

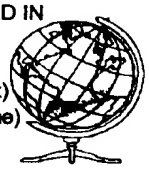
ADD $3.00 S&H FOR FIRST ITEM, $1.00 EACH ADDITIONAL ITEM

SERVICE, PARTS, & HARD TO FIND SOFTWARE WITH COMPLETE DOCUMENTATION AVAILABLE. INKS & REFILL KITS FOR CGP-220, CANON, & HP INK JET PRINTERS, RIBBONS & vr. 6 EPROM FOR CGP-220 PRINTER (BOLD MODE), CUSTOM COLOR PRINTING.

Terry Laraway
41 N.W. Doncee Drive
Bremerton, WA 98311
360-692-5374


```
(Y*32),128:POKE&H401+O+X*2+(Y*32),128
20250 IF C$="1"THENPOKE&H400+O+X*2+
(Y*32),A:POKE&H401+O+X*2+(Y*32),A
20260 NEXT X,Y
20270 O=O+8:IFO/32=INT(O/
32)THENO=O+32*3
20280 IF O >12*32+30 THEN GOTO 20300
20290 GOTO20200
20300 PRINT@14*32,"DO YOU WISH TO
PLAY AGAIN? <Y/N>";
20310 DEST=&H6400
20320 FOR X = &H400 TO &H5FF
20330 POKE DEST,PEEK(X)
20340 DEST=DEST+1
20350 NEXT X
20360 PRINT"BYTEMAP SAVED IN
DATA.BIN"
```


kelly@polar.enet.dec.com (work)
ci245@freenet.carleton.ca (home)



What are you waiting for?

Get your friends to subscribe to the only magazine that still supports the Tandy Color Computer...

"the world of 68' micros"!

The more people who want support, the longer it will be here!

The Embedded Programmer

Paul K. McKneely

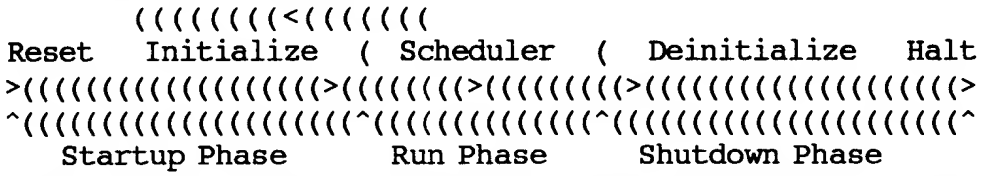
Life and Death of the Boot Process

One of the most intriguing things about advanced computer system software is how it appears to run many programs at once when we know that there is only one processor in the system. This problem becomes much simpler to comprehend when we narrow our focus to one "main program" that is active all the time that I call the Boot Process. Before we go too much further, though, let's define a few terms:

Program: A functional unit of executable code.
Process: A sequence of program execution, including its runtime context.
Task: A kind of pre-emptible process together with its threads and associated I/O resources that does not have system priviledges.
Thread: One stream of execution within a Task.

Various authorities define these terms differently so we will use these rough definitions for the purposes of our discussion. Notice that this definition of the word process is very general and it applies to system as well as application software. In fact, the execution of an interrupt service routine itself is a process. A task, on the other hand, is more specific and it excludes the inner parts of system software. Furthermore, task does not even imply that there is only a single execution stream. In modern operating systems, it is possible for a single task to have many execution streams (i.e. threads) active at once.

If we ignore, for now, that our system may be very complex with many things going on at once, we can get a better idea of how to coordinate such a system using a single process called the Boot Process. When you first apply power to your computer, the processor begins to execute a single stream of execution that is never pre-empted (but will be temporarily suspended by interrupt service routines) and it basically runs until your system either crashes or completes an orderly shut down. The Boot Process is the main process responsible for tying all of the system together and it oversees program execution by the processor. So if we were to draw a simple diagram of the life of the Boot Process it would be something like the one below:



The life cycle of the Boot Process is divided into three phases: Startup Phase, Run Phase, and Shutdown Phase. Startup Phase begins with system reset. There are quite a lot of things that have to get done before the system is ready for use and this is where the basic kernel-level initialization is done. When these things are completed, the Boot Process enters the Run Phase. Here the Boot Process enters a program called the Scheduler which is a looping algorithm that determines what other processes are to get execution time. The only processes that are not directly controlled by this algorithm are the interrupt processes. This is because the mechanism that decides what interrupt processes are to run is done in hardware. When the person responsible for the system decides to shut it down, the Boot Process eventually exits the Scheduler, entering the Shutdown Phase. This causes things to be cleaned up properly so the system can be turned off.

The Startup Phase is the most complicated of all of the phases because it has to get the computer in normal running order. By the time this phase is completed, all of various parts of the Kernel are initialized and ready for operation. Let's look at this phase in a little more detail. Below is a list of the major steps that take place during the life of the Boot Process with the Startup Phase (marked by*) resolved into several steps:

*Reset
*Set up temporary ISP
*Scan Main Memory (and possibly clear)
*Partition Memory
*Initialize Kernel
*Initialize On-board Device Objects
*Load Startup Task
Run Scheduler
Perform Orderly Shutdown
Stop

Reset

Reset is a rather simple step because most of the important work is done in hardware. A well-designed computer will route a system reset signal to all of the various hardware components and get them ready to start from the beginning. The most critical thing that has to happen during reset is that all interrupting devices must turn off their interrupt requests.

Setting Up a Temporary ISP

Most 68K-based computers have some memory available immediately at power-up. It is often dynamic memory or ECC (Error Checking and Correcting) memory that requires initialization before it can be used. Even so, in many systems it is necessary to set up a temporary stack before the rest of initialization can take place. For example, the memory scan algorithm discussed in the next section has the potential of generating an exception so an interrupt stack with at least a few bytes of good memory must be set up. In the CD68X20, this is not a problem because main memory is immediately ready for use. Setting the ISP at 16K is a good place to start. In systems like the MVME177 where the main memory module must first be initialized before use, the board provides an appreciable amount of static memory that can be used before the main memory is ready.

Measuring Main Memory

The first thing our system will do after reset is to find out how much main memory there is. There is little our system can do without main memory at its disposal. In some systems this is a trivial task and in others it can get rather complicated. The (Kernel requires that main memory begins at location 0. This has two advantages. First of all, the original 68000's EVT starts at 0 so we know that our kernel will work with the whole 68K family. Secondly, you can efficiently access the first 32K of address space using short addressing.

For those of you who are used to writing functions in assembly language that are called from HLL's such as C and Pascal, it is worthy to note that the Boot Process does not have to save any of the registers before using them. This is because it is the only code in the whole system that was not called by something else. However, the Boot Process does reserve certain registers to be used for special purposes during parts of the startup sequence. After reset is complete, the A6 register is reserved for pointing to the byte location that is one past the last usable byte of main memory. Notice that this address turns out to be the total amount of usable main memory because main memory always begins at location 0.

The example I will discuss works on the CD68X20 from Peripheral Technology. The CD68X20's main memory is ready for use immediately after reset. All you have to do is to find out how much there is. This is in contrast to the MVME167 and MVME177 single board computers from Motorola that require complicated initialization before their dynamic RAM modules can be used. But even on the CD68X20, detecting the end of memory is more tricky than you might think. Below is a sequence of code that measures the amount of main memory available to the system. For reasons to be discussed later, main memory is assumed to be (and can only be used) in increments of 4 KBytes (Pages):

```
;Measure Main Memory in increments of 4K
;D0=All 0's Bit Pattern
;D1=All 1's Bit Pattern
;D2=Test Register
;A0=Original ISP
;A6=End of Memory
SizMem:   MOVE.L  A7,A0       ;Save ISP
          LEA.L   (SizMem9,PC),A1
          MOVE.L  A1,(8).W    ;Buss Error Vector
          SUB.L   A6,A6       ;A6=End of Main Memory
          CLR.L   D0          ;D0=All 0's
          MOVEQ   #-1,D1      ;D1=All 1's
;Try 1st combination
SizMem0:  MOVE.L  D1,(A6)     ;Write the 2nd Pattern
          MOVE.L  A6,D2       ;At 1st Page?
          BEQ.B   SizMem1
          TST.L   (0).W       ;Wrapped Around?
          BNE.B   SizMem10
SizMem1:  MOVE.L  D0,(4,A6)   ;Write the 1st Pattern
          CMP.L   (A6),D1     ;2nd Pattern Still There?
          BNE.B   SizMem10
          CMP.L   (4,A6),D0   ;1st Pattern Still There?
          BNE.B   SizMem10
;Try 2nd combination
          MOVE.L  D0,(A6)     ;Write 1st Pattern
          MOVE.L  D1,(4,A6)   ;Write 2nd Pattern
          CMP.L   (A6),D0     ;1st Pattern still there?
          BNE.B   SizMem10
          CMP.L   (4,A6),D1   ;2nd Pattern still there?
          BNE.B   SizMem10
;Approved!
          ADD.W   #4096,A6
          BRA.B   SizMem0
SizMem9:  MOVE.L  A0,A7       ;Restore ISP
SizMem10:
```

We begin our algorithm by setting our end-of-memory boundary pointer (A6) to the beginning of memory (location 0). A6 points to the first location within the next Page to be tested. All memory locations lower than the value in A6 (unsigned) are considered to be good memory. We test all bits of the first 8 locations of the page for the ability to store 0's and 1's. If all bits pass this test, the whole page is approved and we go on to the next page. The way we do this is to write all 1's to the first four byte locations then we write all 0's to the next four byte locations. We then go back to see if the 1's we wrote in the first four locations are still there. The reason why we alternate between two patterns instead of using just one at a time is because there can actually be enough capacitance in the data lines to store the values intended to go to memory even if there is no memory there. Since we are testing for 1's when the processor just wrote 0's we can be assured that there is good memory there if the value returned is the same as what was written at that location earlier.

To make sure that we are not just reading back values that the bus circuitry always likes to return at these locations even without memory actually being present, we do the test again but we reverse the patterns being used, still making sure we alternate patterns between reading and writing. If the values still compare, then we know there must be good memory there.

But there are still two other problems that we need to be prepared for. Many 68K-based computers cause bus error exceptions when the processor attempts to access memory that is not present. If this happens, the first attempt to write a pattern beyond actual memory will cause an exception and the next instruction will never be executed. To handle this situation we first set up the appropriate exception vector before our loop begins. Our algorithm assumes that a bus error indicates that

the end of memory has been encountered and the scan is terminated. The last value that was present in A6 before the exception occured is used as the end of memory. If an exception is taken we simply restore the original value of the ISP (saved in A0) effectively throwing the exception stack frame away.

There is a second problem that we must be prepared for while measuring memory. The CD68X20 does generate a bus error but only when accesssing locations beyond the end of memory when the system memory is maxed-out. This means that the use of a bus error exception to determine how much memory is in the system will only work if you have 128 MBytes of DRAM. If your system is like mine, you don't have near that much memory in your computer. What can happen in systems that don't have the maximum amount of memory they were designed to accomodate is that the memory you do have will repeat through the address space that is set aside for main memory. This can cause a problem when you are trying to find out how much memory there is because you might start testing the same memory over again. The way we avoided this problem is by knowing that we end our test loop by leaving all 0's in the first four locations. After writing all 1's to the first four locations of the next page we go back and test the very first four locations in Page 0 if A6 does not point to page 0. This effectively tests for memory wrap-around and if it never occurs, the first four locations will stay 0's like we left them. If memory ever does wrap-around, it should happen first at location 0 and we exit the loop.

Partitioning Memory

Once the Boot Process knows how much main memory is available, it can begin setting it up for use. The (Kernel has a number of different ways main memory is used and it divides memory into five pieces called partitions before the kernel is initialized. These five partitions are listed below in the order they appear in memory:

Absolute
Kernel Image
Dynamic Memory
Interrupt Stack
Page Memory

The size and location of the Absolute Partition is actually decided, not by the Boot Process, but by the linker. This is the only partition that is accessed using fixed locations. Since the Absolute Partition begins at location 0, the kernel can use short addressing to access data stored here. The locations of all of the other partitions are determined at runtime and must be accessed using methods other than absolute addressing.

Following the Absolute Partition is the Kernel Image Partition. The Kernel Image is that part of the code which contains the (Kernel if it is to be loaded into main memory. In some cases, the kernel may be burned into ROM. If this is the case then a significant amount of RAM may be saved by directly running it out of ROM making the Kernel Image Partition optional. Even if the kernel is burned in ROM, often times ROM is slow when compared to RAM so that the Kernel Image may be loaded into RAM anyway just to get more performance. In the CD68X20, the boot ROM is only 8-bits wide and code running from there is quite a bit slower than it would be if running from RAM, so I usually copy the ROM code into a specially prepared Kernel Image Partition when I burn the kernel itself into ROM. If the Kernel Image is loaded from a storage device, such as a floppy drive, then the Kernel Image Partition is certainly necessary.

Next is the Dynamic Memory Partition. The exact location of the first usable byte must fall on an even 16-byte boundary. These 16-byte chunks of memory are called paragraphs and are the basic unit of allocation from this pool of memory that acts much like the heap that is so familiar in the C language.

Next is the Interrupt Stack Partition which remains fixed in size once the memory partitions are set up. Remember that the 68K uses a push-down stack so that the ISP is first set up to point just past the end of the partition. This means that it starts out pointing into the first byte of the next partition which is the Page Memory Partition.

Finally, the Page Memory Partition is useful for storing data for block-oriented devices such as a disk drive. This partition is always a multiple of 4 KBytes and is divided into 4 KByte Pages for allocation. Because memory is measured in multiples of 4 KBytes and because the Page Memory Partition is the last partition, each 4 KByte Page is guaranteed to begin on an even 4 KByte boundary. This is the memory pool from which virtual memory is implemented.

Initializing the Kernel

After memory has been partitioned, the kernel can be initialized. Without going into too much detail about how the (Kernel works it should be said that it is composed of modules. Some modules use others so that the different modules must be initialized in order of dependency. This ensures that services that are required for the initialization of each module are already in place before they are needed.

Initializing Onboard Device Objects

Almost all SBCs come with a number of I/O devices built-in. As with the PC's BIOS, it makes sense to include a device driver in the Boot ROM with each specific I/O device that is part of the SBC. If your system is completely embedded then this is relatively straight-forward. But this is a problem if your Kernel Image was designed to be loadable so that it can run unchanged on a number of different computer systems. When your Kernel Image is loaded from an external storage device you would like it to be able to make use of all onboard devices and run correctly. The loadable image should not have to be specially prepared for the particular computer you are running it on. In such a case, you will want your Boot ROM to contain a list of device drivers that will not actually be installed by the code in the Boot ROM. A pointer to a table describing these device drivers is passed to the Kernel Image after it is loaded from disk. The loaded image can then install the device objects using the kernel facilities that came with the Kernel Image.

Loading the Startup Task

Once your kernel is in place, you will not want to do any of the user-specific initialization in the kernel itself. This kind of work is best done using a task whose program is loaded from either the Boot ROM or from a storage device. Once the task has been initialized and is ready to run, the Boot Process can enter the Scheduler. The Scheduler will then see that it has a task that is ready to run and it will give the task processing time.

In the next article we'll look deeper into memory partitioning, PREBOOT, and how a basic Scheduler works. If you have any comments or requests, please feel free to write me at either <gecko@onramp.net> or at the address given below:

Paul K. McKneely
technoVenture, Inc.
P. O. Box 5641
Pasadena, Texas 77508-5641

CoCo3 Extended Memory Secrets Part 5

Herbert Enzman

Add menu and text windows to your programs

In this part, I'll show you how to add a little "pizzaz" to your programs with the addition of Menu and Text Windows. Although they are not 'TRUE' multi-tasking 'windows', they can add some interest to dull screens. These windows will write over the tops of screens and each other, but when you exit the window, the original screen will still be there. LISTING 15 is the DEMO program for this part, and is far from complete; but it is a building block that you can use your imagination with. To demonstrate it's capability, I've only included one workable entry in each menu. This keeps the program as short as possible. After you see how it works, you can play with it; move windows around, try your own, etc.

So, let's get started. Type in listing 15, save it to disk. Since the program is a little long, it will be best to assemble it to disk and then load it from Z-BUG to run. When it is assembled, go to Z-BUG and load it into memory; run it with "GGO" and follow along with the program "play by play".

After starting it up, you will see a menu overwrite Z-BUGs screen, which will be the "UTILITIES" menu. Using the UP/DOWN arrow keys, move the cursor thru the menu. You will notice that the 'cursor' is just a different color for the text.

I chose this method to be different. The old 'reverse video' from the COCO 1 days was getting boring, and I thought that something different was in order.

Now move the cursor down to the entry "disk routines" and press <ENTER>. This is the only active entry for this menu; all other choices will exit the 'utilities' menu. You will then see a second menu appear on the screen. Press the <?> key and this will demonstrate another use for the windows — a help screen. It is rather small, but can be made to any size. Press <?> again and you will see a second help screen. To exit back to the 'Disk Routines' menu, press <BREAK> twice (once will take you back to the 1st help screen).

Now using the UP/DOWN arrow keys again, choose the entry "Clean Drive". You will be greeted with a third Menu. Choose a drive number and press <ENTER> and you will see a 'text' window pop up with a graph in it. This is a routine that will run the drive motor for 30 seconds to clean the drive, while the graph counts off the seconds. When it is done, the text window will be erased and you will be back to the 'select drive' menu. Press <BREAK> to exit back to the 'disk routine' menu, and select the last entry to exit this menu. Select the last entry again to exit back to Z-BUG, and you will find the MAIN screen just as you left it.

See what I mean by adding "pizzaz" to your programs? And it is all done on the text screen. No need to gobble up large amounts of memory by using graphics screens. Even though the listing is commented, there are still a few things that will need discussion. If you are still interested, just continue.

I'll start with the LABEL "MENU1", because it is used by you to tell the program how to set up a window. You will need the screen grid that you made during PART 2 of this series to locate screen addresses. To aid in the layout of the windows, you can overlay a blank sheet of paper on top of the screen grid.

The first thing you want to do is figure out what you want in the menu. How many entries, where you want extra text (such as headings or special instructions) and what the entries will be. Write down the menu entries and pick the longest text entry and count it's number of characters. This will tell you how wide the window should be. The number of menu entries, extra text and extra 'space' lines will tell you how long the window will be.

With this information in hand, you can now choose where you want to put the upper left corner of the window. Put this address in the 1st 'FDB' location. See TABLE 15 if you want to use a DECIMAL format instead of HEX. They both work the same, but they have to be different for the assembler to assemble the decimal numbers properly. The one in LISTING 15 is for HEX only.

Now you want to tell it how many characters (including left and right margins) wide you want the window and how many lines down that you want it. This information goes into the next FDB (or next 2 FCB's if decimal format). Next choose the windows background color, and tell it how many MENU entries that there are (don't include headings, etc.). This goes in the 3rd FDB. The 4th FDB is used to tell the routine what color to use for the menu text and highlight colors. Make sure the highlight color stands out from

TABLE 15 - Alternate format for Menu / Windo tables for DECIMAL use.

```
MENU1 FDB   8520   0,U   Upper left corner of window Address
      FCB   26     2,U   # of characters per line
      FCB   19     3,U   # of lines in window
      FCB   10     4,U   Window color
      FCB   10     5,U   # of menu entries
      FCB   42     6,U   Highlight color
      FCB   58     7,U   Menu text color
      FDB   9008   8,U   1st menu entry start Address
      RMB   2      10,U  Lower cursor limit
      RMB   2      12,U  Cursor TEMP
      RMB   2      14,U  down counter / (15,U) up counter
      RMB   10     16,U  Attribute table for menu 1


WINDO1 FDB   8584 0,U   Upper left corner of window Address
       FCB   34   2,U   # characterrs per line
       FCB   7    3,U   # lines in window
       FCB   5    4,U   window color
```

the rest of the text and also goes with the background color. Remember, the highlight color will be the cursor.

The last thing to choose is where you want the 1st menu entry to start. After telling the routine this, it will fill in the other entries in thier proper locations. The 3 "RMB 2" entries will be filled in by the routine. The last 'RMB' reserves locations for attribute table for the menu entries and should reflect the same amount as the number of menu entries (10 in the 1st menu).

Text windows need the least amount of information, as can be seen in the label "WINDO1". Again, see TABLE 15 for the Decimal format. Basically, you are just putting the window on the screen, and then filling in the text that you want. The "HELP" screens will demonstrate the keyboard poll routine to use. The "GRAPH" text window is the most comlicated, because it has more to do. Most text windows can be simple, for example, the help windows, or for error messages, etc.; let your imagination run wild!

Looking at LISTING 15, you will notice that each Menu routine is fairly similar, with the exception of the extra text, headings and keyboard decode routines.

The "GETRDY" routine points to the attribute table and gets the proper color code for the menu entry that it is going to write. The "CALC" routine will calculate the routine address from the offset table for that particular menu.

The "DWNARW" and "UPAROW" routines are common to all menus. It will take care of the cursor location, moving the cursor and the up/down counters. The "TUBE" routine will fill in the menu entries for each menu.

The "SETATT" routine is only used on the 1st entry to a menu. It is responsible for setting the attribute codes for the menu entries. It is only used on the start up of each menu, because the cursor will be where you left it when you return to that menu from another. This way, it won't always start at the top of the menu.

The "KWIK" routine is the keyboard check routine used by all menus and text windows. It will swap the task register so that the ROMs can be used (remember that the ROMs are located in TR=0). The "STRING" / "SCRIPT" routines are used to write text to the screen. Enter "STRING" when the text attribute is located before the text and "SCRIPT" if you want to use a text line that exists with a different color.

The "SETUP" routine is responsible for saving the screens before it puts a window on the screen. It is why the previous screen(s) are still intact when you return to them. Looking at the listing, you will see that before doing a window, register 'B' is loaded with a 'positive' number (1 in the Utilities menu), which tells the "SETUP" routine to save the 1st screen. When the utilities menu is done it sends $81 to the setup routine to tell it to restore the original screen. The table "START" at the end of the listing is used by this number (1-3) to determine the start address of the screen to save or restore screens, depending upon the level of windows that are on the screen. For example: "Utilities" uses level 1 ($81 for restore); "Diskit" uses level 2 ($82) and "Clean" uses level 3 ($83). This way, every time you write over the top of a screen with a new menu, this routine will know what level that was used and will put back the previous screen. It sounds complicated at first, but you will see what I mean when you play with the program.

The "WINDOW" routine is called to put the window on the screen with the parameters from the 'MENU' tables. It does not put any text on the screen, just the window. The final subroutines "CLEAR" and "SET", are used to swap the task register to access the ROMs.

"UTLJMP" and "DSKJMP" are the routine jump tables for the first two menus. The address is in the form of an offset, hence the format "FDB SPARE-GO". To add your routines to the jump table; just replace the first word (SPARE in the UTLJMP table) with the name of your routine. This way, the "CALC" routine will point to your routine, no matter where it is.

"TXTAB1", "TXTAB2" and "DRTXT" are used for the menu text. Use this format. It uses the 'negative' stop character format, so the last character should be negative so the 'screen' routine will know when to stop printing a line of text. The same is true for "TXT1" thru TXT13". The "FCB" before each "TXT" line is used for the color of the text that follows it (see the comment column for 'TXT1').

LISTING 5 - Write to screen routine

```
 UPON ENTRY:
X = Points to text that is to be printed to screen
Y = location on screen to put text
B = SEE TUTORIAL TEXT


STRING LDB   -1,X     get attribute that is before text string
SCRIPT LDA   ,X+      get text character
       PSHS  A        save for stop character test
       BSR   FIX      send it to the screen
       TST   ,S+      was character a STOP char. (negative)?
       BPL   SCRIPT   no, loop for more characters
       RTS            DONE - return
FIX    ANDA  #$7F     drop MSB first
SCREEN PSHS  D        save for after block swap
       ORCC  #$50     disable interupts
       LDA   #$36     = BLOCK # that scren uses
       LDB   $FFA3    *** $FFAB for 6309 users
       STB   SAVE     save current block # for return
       STA   $FFA3    *** $FFAB
       PULS  D        get character and it's attribute
       STD   ,Y++     store both to the screen
       LDB   SAVE     get original block # that was saved
       STB   $FFA3    *** $FFAB
       ANDCC #$AF     enable interupts
       RTS            return for more characters

SAVE   RMB   1        ** temp for current block #

       FCB   XX       PUT desired attribute here in place of
'XX'
TEXT1  FCC   /YOUR MESSAGE HER/
       FCB   '-+$80    this is for STOP printing code
       END
```

LISTING 6

```
GO    LEAX  TEXT1,PCR
  LDY   #$2AF0
  BSR   STRING
  SWI
```

LISTING 7

```
GO    LEAX  TEXT1,PCR
LDY   #$2AF0
LDB   #$XX   = attribute
BSR   SCRIPT
SWI
```

LISTINGS 6 AND 7 call LISTING 5. — SEE TEXT

LISTING 8 "Second" screen demo

***** CODE with "*" is for E/A 6309 users only!!!!!!!

```
GO    NOP
*     CLR   $FF91   set TR=0
      LDA   #$3F    end address of second screen
                    (msb)
      STA   $FE06   set it for SECB
      STA   $F688   set it for SECB
      DECA          'A' now = $3E
      STA   $F875   set SECB
      LDA   #$30
      STA   $F7BC   set start address of second screen
                    (msb)
      STA   $F68D   set SECB
      STA   $F6A3   set SECB
      STA   $F6D5   set SECB
      JSR   $F679   Now set up 80 column screen
*     LDA   #1      ###
*     STA   $FF91   ### set TR=1
      LDD   #$3600  set screen colors
      STA   $FFB8   set foreground to yellow
      STB   $FFB0   set background to black
                    (disk edtasm)
*     STB   $FFB4   set background to black
                    (e/a 6309)
      STB   $FF9A   set border to black
      SWI           FINISHED
      END
```

That's pretty much all that there is to it. Now that you have an idea of how this routine works; you can play with it. Change the window positions, colors, # of entries, window sizes, etc. Add some routines of your own, and add them into the 'JUMP' tables to see how that works.

In the final part (PART 6) of this series, I will show you how to load everything on program start-up, even if a program is in a block that is different than the current task register set-up.

VIDEO GAME MAGAZINES WANTED!

Video Games Player, Atari Age, Electronic Fun, Electronic Games, Replay, Playmeter, Blip, and Atarian. Write with what you have and what condition they are in and I'll send back a quote. Video Games, Box 9542, Pittsburg, PA 15223

RGBoost - $15.00

If you want to speed up DECB easily, install an Hitachi 6309 and get RGBoost. This patch for DECB uses the extra 6309 functions for up to a 15% gain in overall speed. It is compatible with all programs tested to date! Save an additional $5 by purchasing RGBoost along with one of my other products listed below!

EDTASM6309 v2.02 - $35.00
Patches Tandy's Disk EDTASM to support Hitachi 6309 codes! Supports all CoCo models, including stock 6809 models. CoCo 3 version uses 80 column screen, runs at 2MHz. YOU MUST HAVE A COPY OF DISK EDTASM. This is a PATCH ONLY! It will not work with "disk patched" cartridge EDTASM

CC3FAX - $35.00
Receive and print weather fascimile maps from shortwave! The US weather service sends them all the time! Requires 512K CoCo3 and shortwave receiver. Instructions for simple cable included.

HRSDOS - $25.00
Move programs and data between DECB and OS-9 disks! Supports RGB-DOS - move files easily between DECB and OS-9 partitions! No modifications to OS-9 modules required.

DECB SmartWatch Drivers - $20.00
Access your SmartWatch from DECB! Adds function to BASIC (DATE$) for accessing date and time. *Only $15.00 with any other purchase!*

Robert Gault
832 N. Renaud
Grosse Pointe Woods, MI 48236
313-881-0335
Please add $4 S&H per order

Emulator Transfer Tricks

continued from page 3

What you have done is simple in concept. The OS9 disk transfer utility creates an image of the 3.5" disk on the PC's harddrive. However, this image is of a double-sided disk and the emulator will only handle single-sided disks. Therefore, the disk editor is used to change the disk image's format information to say it is single, not double sided. That's all there is to it.

Final Notes

Despite the fact that you have changed the format information stored on the disk image to indicate a single-sided disk, the format information stored in the drive descriptor used to access the image must match the format of the original disk or the emulator will not read the image properly.

I have presented this transfer method in terms of the double-sided 3.5" disks since that was my major concern. However, it would probably work just as well with double-sided 5.25" disks. Also, while I used the dEd disk editor under OS9 to modify the disk image, you could probably do the same thing under MS-DOS using a disk editor like that in Norton Utilities. This could be particularly useful for those people who are just getting started with OS9 on the emulator and trying to transfer their OS9 boot disk.

The only other way to transfer double sided 3.5" disks is obvious: copy as much as possible to a single-sided disk, transfer, then copy the remaining files and transfer them also. This is time consuming but may be your only choice if the tools mentioned are not available.

A Change of Directory

Mark Heilpern

A "Change Directory" command for OS9/68K (modify for 6809?)

Here is a program I wrote quite a while back to implement a stand-alone 'cd' utility. It gets around the memory protection problem by calling _os_permit() to get access. For this to work the program must execute as super-user (group 0). If you will not always run this as super-user you must modify the code somewhat (make the module owned by group 0 and toss in a _os_setuid() to change yourself to group 0 early in the program). If you have no MMU or are not running the SSM extension there is no memory protection.

Questions? I can be reached via e-mail at: heilpern@microware.com. If you don't have e-mail access, feel free to write the editor in reference to this article.

This code IS NOT guaranteed to work, but did the last time I compiled it.

```
/*** This is the first of 2 files ***/
#include <types.h>
#include <stdio.h>
#include <process.h>
#include <errno.h>
#include <cglob.h>
#include <modes.h>
error_code find_proc_desc(process_id, procid **);
main(u_int32 argc, char **argv)
{
   char *pathname = argv[1];
   u_int32 mode = S_IREAD;
/* change data directory */
   procid *me, *dad;
 /* if no directory was specified, check for default */
   if (argc==1) pathname = (char*)getenv("HOME");
   if (pathname==NULL) exit(E_BPNAM);
/* check for execution directory change request */
   if (argc>2)
   {
      if (!strcmp(argv[1],"-x")) pathname = argv[2];
      mode = S_IEXEC;
/* change execution directory */
   }
/* do the directory change */
   errno = _os_chdir(pathname,mode);
   if (errno) exit(errno);
 /* find my process descriptor */
   errno = find_proc_desc(_procid, &me);
   if (errno) exit(errno);
/*find my parent's process descriptor */
   errno = find_proc_desc(me->_pid, &dad);
   if (errno) exit(errno);
/*copy over the directory information */
     errno = _os_cpymem(_procid,&me->_dio,&dad-
>_dio,DEFIOSIZE);
   if (errno) exit(errno);
/* and exit */
   exit(0);
}

/*** the next file is for the find_proc_desc() function ***/
#include <process.h>
#include <sysglob.h>
 /* to get system globals */
#include <stddef.h>
/* for 'offsetof() macro */
#include <modes.h>
 /* for permit access modes */
#include <errno.h>
extern process_id _procid;
/* my id */
/*
** Usage:
**      process_id proc_id;
**      procid *proc_desc;
**        errno = find_proc_desc (proc_id,&proc_desc);
*/
error_code find_proc_desc(process_id proc_id,
   procid **proc_desc)
{
   u_int32 *ptab;
   u_int32 size;

   /* first, find the system's process */
   /* database table */
                              (void)_os_getsys((offsetof
(sysglobs,d_prcdbt)),sizeof(u_int32*), (glob_buff*)&ptab);

/* get access to this memory region */
/* number of bytes we need access to */
/* (size) is the process id of interest, */
/* times size of each pointer entry (4) */
/* plus the size of one entry (4) */
   size = (proc_id+1)*4;
   errno = _os_permit(ptab,size,S_IREAD,_procid);
   if (errno) return(errno);
/* got the table. lets index into it */
   *proc_desc = (procid*)ptab[proc_id];
/* finally, get access to that memory */
   (void)_os_permit(*proc_desc,sizeof(procid),
      S_IREAD|S_IWRITE,_procid);
/* note, don't need error checking on */
/*the last _os_permit(),since the call */
/*should only fail if not running */
/*as super user. since the first permit */
/*call worked, we must be a SU */

   return(0);
}
```

NEW Hardware coming from Cloud Nine!

512k - 2048k upgrade board

Just install SIMM memory in 512k increments (2x256K 8 or 9 chip SIMMs). Three chip SIMMs WILL NOT work! This is a timing requirement, as the 8/9 chip SIMMs use the same timing as the CoCo DRAM upgrades.

SCSI Host adapter interface

- Comes with OS9 Drivers, 6x09. 63b09e 1.78MHZ system "megaread" times are ~11 seconds with 512 byte sectors (Nitros 2.00 Level3).
- 256/512/1024 Sector size selection
- FULL SCSI ID supported
- Parity generation, enable/disable. Can use with parity devices such as ZIP drives!
- Gold plated card edge connector
- 50 pin SCSI header port
- Installation/Operation Manual
- Schematic package
- OS9 Utilities SCSItools, SCSIdesc, ZIP/JAZ Tools
- SCSItools - A BASIC09 utility that will do low level SCSI commands.
- SCSIdesc - A BASIC09 utility program that will create the SCSI descriptor for you based upon the menu drive options inputted.
- ZIPJAZtools - This utility will aliow the features of the Iomega ZIP/JAZ drives. Eject disk, software protection are some. This utility isn't written yet, but I have the documentation needed from Iomega. Will do this soon!

These products should be available at the Chicago CoCoFest! Look for me there!!

A 512K SIMM upgrade is ready to ship. The unit will ship with the following items:

1 - 512K SIMM Memory Board with 8 or 9 chip 120ns or faster SIMMS

1 - Installation Manual

1 - Schematic package

1 - RSDOS Memory Test Program supplied on 5 1/4" disk.

$40 each including shipping, UPS ground, within the US. If you are outside of the US please indicate method of shipment desired and I will check into the added cost, if any.

Cloud Nine
c/o Mark Marlette
3749 County Road 30
Delano, MN 55328
email : mmarlett@isd.net
voice: 612-972-3261

Emulator Sand Patch

Robert Gault

Patch "Sands of Egypt" for the CoCo Emulator

A while back I tried out Jeff Vavasour's emulation of the Coco 1&2. Definitely and interesting package.

One of the things I checked was how well the emulator handled color artifacting and programs like "Sands of Egypt." The emulator does the best possible with an RGB monitor system. It switches to a fake PMODE3 format using the colors white, black, red, and blue. This does not produce the intermediate artifact colors of yellows, greens, and purples, but there is no better practical technique.

"Sands of Egypt" is another matter. This program makes use of an unusual copy protection scheme which fails with a typical PC disk controller. The program looks for a sector which does not have data address marks and quits if the entire disk is good. PC controllers (at least with Jeff's emulator) do not give errors with missing data marks and "Sands" won't run. That is to say, "Sands" goes through the color check test and then immediately says, "this adventure is over."

However, all is not lost for this great Coco program! Using the built in Debugger from the emulator, I have found how to patch the "Sands" disk so that the program will run using the emulator. This will also let you copy the program onto a PC hand drive as well as running it from the original floppy.

For this game to work with an emulator on a PC you need 3 patches. Change the following tracks, sectors and bytes:

track	sector	byte	old	new
18	16	1A	27	26
22	5	E4	26	21
23	8	6D	26	21

The track and sector values are in decimal. The byte locations and values are in hexadecimal.

For comment, I can be reached in care of this magazine or via e-mail:

robert.gault@WORLDNET.ATT.NET

BLACKHAWK ENTERPRISES

New Products!

- Data Windows - $69.95 - A complete flat database program for OS-9/68K. Facilities include database creation, searching, maintenance and report generation. By Alpha Software Technologies.
- GNU TWO - $49.95 - This package include a new port of GNU M4, and the AUTOCONF automatic configuration macros. Together with the included port of BASH these tools make automatic configuration of software a much easier chore. Widely used on UNIX and other operating systems, use it now on your OS-9 platform! Includes two new manuals totaling about 110 pages.
- Model Rocketry Tools - $15 - Includes ports of tools for modeling and tracking the performance of various configurations of model rockets. Essential tools for those interested in designing rockets or achieving specified altitudes. Should run on any OS-9/68K machine.

MM/1, MM/1a and MM/1b hardware and other software still available, inquire!

P.O. Box 10552 • Enid, OK 73706-0552 • (405) 234-3911

CoNect

1629 South 61st Street
West Allis, WI 53214
(pulland@omnifest.uwm.edu)
414-328-4043

Fast232- 16550 does serial! Port speed to 115200bps, transfers up to 5000 cps. Addressable to four locations. With OS9 and Nitros9 drivers. **$79.95**

2nd Port Daughter Board - $45.00

OS9 lvl2 *lvl1 available!*

Level2 Bundle	$49.95
os9, b09, mvue, more! plug-n-go for 6809	
Dynacalc+Pgraph	$19.95
Profile	$19.95
TSEdit/Word+vi patch	$12.95
Epyx TriPak	$14.95
Koronis Rift, Rescue Fractulus, Rogue	
King's Quest 3	$9.95
Microscopic Mission	$4.95
Sub Battle Simulator	$4.95

Hardware

64K upgrd 2 or 4 chip	$5.95
512K upgrd *(used)* 0K	$24.95
512K	$44.95
decb1.1rom + manual	$12.95
mpi upgrd sat. board	$9.95
cable, cassette	$5.95
cable, printer	$5.95
cable, rs232 (100ft!)	$19.25
colr mouse (1 button)	$9.95
mono composite monitor *(used)*	$24.95
Orchestra90cc Pak	$12.95

DECB

Disk EDTASM *(used)*	$19.95
Disk ProFile *(used)*	$12.95
One on One	$7.95
Sands of Egypt	$7.95

ROMPaks too! (Inquire for titles)

Parts *(many more in stock!)*

1488/89	.75	68b09e	6.95
1723	1.95	6821a	3.95
1773	6.95	SALT	2.25
2764	2.95	74*6	.35
6802	3.50	74ls133	.42

I've also been working on some ***NEW*** hardware that may be available later. One of these items is a revision of my Expander idea that actually works on most CoCo 3's, not just the occasional "right" one. I'll keep everyone posted on any progress!

Check with me for complete disk drive systems, misc. hardware items, hardware repairs, and hard to find new and used CoCo software not listed!

Shipping & Handling $4 US, $6 Can/Mex, $10 World
offworld destinations please consult local Postmaster!

StrongWare

Box 361 Matthews, IN 46957 Phone 317-998-7558

CoCo 3 Software:

Soviet Bloc	$15
GEMS	$20
CopyCat	$5
HFE- HPrint Font Editor	$15

MM/1 Software:

Graphics Tools	$25
Starter Pak	$15
BShow	$5
CopyCat	$10
Painter	$35

ADVERTISER'S INDEX

BlackHawk Enterprises	*19*
Chicago CoCoFest	*4*
CoNect	*BC*
FARNA Systems	*5, 11, BC*
Robert Gault	*17*
Hawksoft	*11*
Dennis Kitsz	*7*
Nickolas Marentes	*7*
Small Grafx	*11*
StrongWare	*BC*
Video Games	*17*

What are you waiting for?

Get your friends to subscribe to the only magazine that still supports the Tandy Color Computer... *"the world of 68' micros"*!

The more people who want the support, the longer it will be here!